7E−4

HyperStation: InterViews を用いた HyperShell 共有オブジェクトの考察

羽根 秀宜, 岩崎 未知, 新 淳
NEC C&C システム研究所

1 はじめに

我々は、オブジェクト指向マルチメディアシステムとしての HyperStation[1]、及び、この上の分散ソフトウェア環境としての HyperShell[2] の研究開発を進めている。

HyperShell は、複数の利用者がマルチメディア情報を含むさまざまな情報を作成/操作/共有する事を目的とし、その操作対象として分散共有オブジェクトを取り扱う。特にオブジェクトの共有化を考える際に、共有されたことにより実現される機能が、グループウェアなどのアプリケーションを構築する為に柔軟に利用できるような形で定義されていなければならない。

そこで本稿では、HyperShell におけるオブジェクトの共有化の方法と、それをオブジェクト指向ツールキット InterViews[3] を利用して構築する方法について報告する。

2 HyperShell で扱うオブジェクト

HyperShell でのオブジェクトの構成は、図1に示すように親子関係を持つ階層構造をしており、紙と本のメタファに基づくオブジェクト構成になっている [2]。構成しているオブジェクトは本のメタファを実現する Binder、紙のメタファを実現する Layer、Sheet、紙の上に載せる Viewer、Drawing、Field、Button である。特に Viewer は、Binder を「覗く/操作する」ためのオブジェクトであり、利用者は Viewer に Binder を attach することにより、Binder をはじめとするオブジェクトを操作することができる。

この Binder を Viewer に attach することにより、利用者はその Viewer を通して Binder を操作することができる。また、複数の Viewer に一つの Binder を attach することで、Binder の共有を行うことができる。

図1のように Viewer はオブジェクトの階層中にも存在することがある。これは、Binder の階層中での Viewer に attach されているオブジェクトが、更にその Binder を attach されている Viewer を通して操作されている場合であり、本方式では後述するようにオブジェクトの共有の一形態を与える。

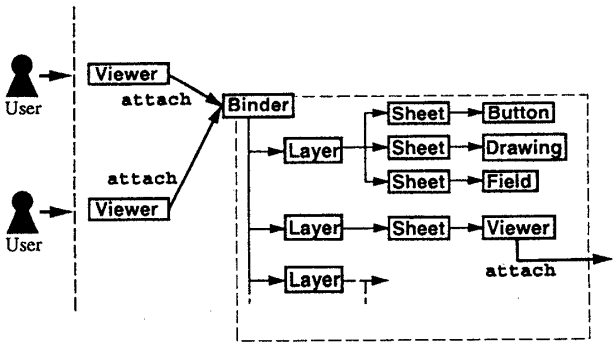

図 1: HyperShell の Binder オブジェクトの構成

3 Binder の共有化方式

共有化を考える際に問題となるのは、1) 何を共有したいのかと 2) 共有する際の単位は何かである。これらは、そのアプリケーションや目的とする動作によって異なるものである。HyperShell では、Viewer に Binder を attach し、Viewer を通して Binder を操作することを基本とすることから、Binder の Viewer に対する描画やイベント処理の意味において、共有、非共有の切り口を考える。つまり、上記の 2) に関して Viewer オブジェクトを単位として共有を行うとする。

その基本的な方針は次のようになる。

- Binder の持つ情報を共有されるものと、共有されないものに分ける。共有されない情報は、Binder を Viewer に attach した際に、その Viewer に対応する情報として確定される。この共有されない情報を、その Viewer に対応するコンテキストとする。
- Binder の持つコンテキストは、その Viewer に対応する、Binder の描画位置や表示している Layer というように、あらかじめ定義されている。

このように、Binder の持つ情報の内、操作している Viewer 毎に、コンテキストが共有されていない情報として存在し、それ以外の情報が Viewer 間で共有される情報として存在することになる。

Binder 内のオブジェクトの描画やイベント処理は、このコンテキストにおいて行われる。したがって、例えば複数の Viewer に attach されている Binder では、

HyperStation: Shared Objects with InterViews in the HyperShell
Hidetaka HANE, Michi IWASAKI, Atsushi ATARASHI
C&C Systems Research Laboratories
NEC Corporation

コンテキストとして現在表示している Layer の番号を持つことにより、それぞれの Viewer において異なった Layer(つまり異なったページ) に対して表示や操作を行うことができる。

4 Viewer の共有化方式

Viewer は、更に複数の Viewer に共有されることがある。

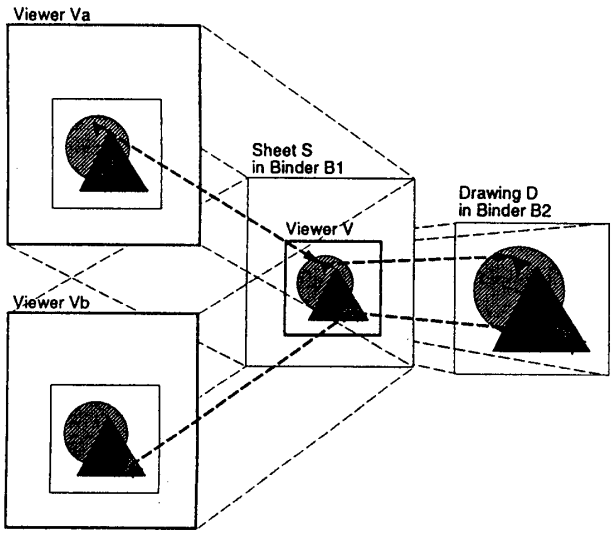

図 2: Viewer の共有

図2では、Binder B1(以下 B1 とする) 内に Sheet S が存在し、Sheet S 上に Viewer V(以下 V とする) が存在する。更に、V に Binder B2(以下 B2 とする) を attach しており、B2 内には Drawing D(以下 D とする) が存在している。

V 上には B2 内の D が表示されている。この場合 B2 は V に対応するコンテキストで描画やイベント処理を行う。

ここで、B1 を 2 つの Viewer Va、Vb(以下それぞれを Va、Vb とする) に attach している場合を考える。この場合、V が Va、Vb に共有されていることになる。そして、B2 が V に描画した D の内容と同じものが Va、Vb に対して描画される。また Va、Vb で発生したイベントが V に伝えられると、V におけるイベントとして発生座標などの変換が行われた後に、V から B2 に伝えられ D に対して処理が行われる。この機構により、V に対応するコンテキストでの Binder B2 内の D を共有することが可能となる。

このように Viewer の共有により、同じ編集対象を複数の利用者から共同して作業を行う形態の共有形態を実現できる。

5 InterViews による実現

InterViews では、ユーザ・インタフェースにおける「見た目」を持つものを Glyph クラスとして定義している。この Glyph クラスは描画やイベントヒットを行うインタフェースと、親子関係を持つ Glyph の階層構造を成すためのインタフェースを持っている。

HyperShell のオブジェクトでは、描画やイベント処理に Glyph のインタフェースを利用する。Glyph が親子関係を持つことは、階層構造を持つ HyperShell のオブジェクト構成とも親和性がよい。

このような観点から、HyperShell での実現方法は以下のようになる。

1. HyperShell のオブジェクトでは、描画などのメソッドは対象となる Viewer に対応するコンテキストにおいて実行する必要がある。したがって、メソッド実行時のコンテキストを指定できるように拡張する。
2. Glyph では、基本的に親 Glyph によって自身の位置が管理されるが、HyperShell ではコンテキストとして管理されるように拡張する。
3. Glyph のインタフェースでは自身の親を知る手段がなく、HyperShell におけるイベント処理などで不都合が生じる。したがってオブジェクトの親を知るためのインタフェースを拡張する。

6 まとめ

HyperStation の分散オブジェクト指向シェル HyperShell における、共有オブジェクトの共有の方式について述べた。

最後に、本研究の機会を与えて下さいました NEC C&C システム研究所の山本所長、小池部長、川越課長に深謝致します。また、有益な助言を下さいました同所の濱川主任、坂上氏、岡本嬢をはじめとする皆様方に感謝致します。

参考文献

[1] 濱川 礼, 他, 「分散オブジェクト指向マルチメディアシステム HyperStation — その構想と試作 —」, 情報処理学会第 45 回全国大会, 1B-01, Oct, 1992.

[2] 新 淳, 他, 「HyperStation: 分散オブジェクト指向シェル HyperShell」, 情報処理学会第 45 回全国大会, 6Q-06, Oct, 1992.

[3] Linton, M., et al, "InterViews Reference Manual Version 3.1", Stanford University, 1992.